Graph Kernels for Chemical Informatics

Liva Ralaivola [a,b], Sanjay J. Swamidass [a,b], Hiroto Saigo [a,b], and Pierre Baldi [a,b,*]

[a] *School of Information and Computer Science, University of California, Irvine CA 92697-3425*

[b] *Institute for Genomics and Bioinformatics, University of California, Irvine CA 92697-3425*

Abstract

Increased availability of large repositories of chemical compounds is creating new challenges and opportunities for the application of machine learning methods to problems in computational chemistry and chemical informatics. Because chemical compounds are often represented by the graph of their covalent bonds, machine learning methods in this domain must be capable of processing graphical structures with variable size. Here we first briefly review the literature on graph kernels and then introduce three new kernels (Tanimoto, MinMax, Hybrid) based on the idea of molecular fingerprints and counting labeled paths of depth up to d using depth-first search from each possible vertex. The kernels are applied to three classification problems to predict mutagenicity, toxicity, and anti-cancer activity on three publicly available data sets. The kernels achieve performances at least comparable, and most often superior, to those previously reported in the literature reaching accuracies of 91.5% on the Mutag dataset, 65-67% on the PTC (Predictive Toxicology Challenge) dataset, and 72% on the NCI (National Cancer Institute) dataset. Properties and tradeoffs of these kernels, as well as other proposed kernels that leverage 1D or 3D representations of molecules, are briefly discussed.

Key words: kernel methods, graph kernels, convolution kernels, spectral kernels, computational chemistry, chemical informatics, toxicity, activity, drug design, recursive neural networks

* Corresponding author.
Email address: pfbaldi@ics.uci.edu (Pierre Baldi).
URL: www.ics.uci.edu/ pfbaldi (Pierre Baldi).

1 Introduction: Processing Structured Data

Many problems in artificial intelligence, data mining, and machine learning involve variable-size structured data, such as strings and sequences, trees, and directed or undirected graphs. These data are to be contrasted with the standard, fixed-length, vectorial data found also in many applications. Examples of structured data include: (1) text and documents in information retrieval; (2) DNA/RNA/protein sequences and evolutionary trees in bioinformatics; and (3) molecular structures in chemical informatics. Clearly the structures present in structured data are essential for the more-or-less automated extraction of meaning, patterns, and regularities using machine learning methods. Hence, it is important to develop general inferential methods that can handle structured data of variable sizes and dimensions. Here we focus on graph-structured data and the development and application of kernel methods for graph-structured data, with a particular emphasis on chemical informatics and the prediction of the toxicity or biological activity of chemicals.

Broadly speaking, several classes of general machine learning methods have been developed to process structured data, including molecular data. A somewhat arbitrary and non-exhaustive classification includes: (1) inductive logic programming (ILP); (2) genetic algorithms, and other evolutionary methods; (3) graphical models; (4) recursive neural networks; and (5) kernel methods.

The basic idea behind ILP [Muggleton, 1992] is to represent a domain and the corresponding relationships between data items in terms of first-order logic predicates and to learn logic theories from data via a process of induction, i.e. an ordered search of the space of possible hypotheses. ILP has been applied with some success to chemistry problems, in particular to QSAR [King et al., 1995] and mutagenicity prediction [Muggleton, 1992, Srinivasan et al., 1996] (see also Chou and Fasman [1978] for a related early application to protein secondary structure prediction). The ILP approach has several attractive features: (1) it can handle symbolic data in a natural way; (2) background knowledge can easily be incorporated into the rules; and (3) the resulting theory and set of rules are relatively easy to understand. The main drawback, however, is the lack of efficiency. Even on current computers, the induction/learning phase is inherently hard because the space of possible hypotheses/theories on realistic datasets is extremely large. Stochastic grammars [Sakakibara et al., 1994] can be viewed as a family of related approaches, also related to graphical models. The simplest stochastic grammars, e.g. stochastic regular grammars and the related hidden Markov models, come with efficient learning algorithms. However, more complex stochastic grammars suffer from computational complexity issues similar to ILP methods.

The basic idea behind evolutionary methods in general, and genetic algorithms

[Koza, 1994] in particular, is to evolve populations of structures, or programs specifying structures, using operators that simulate biological mutations and recombinations, together with a filtering process that simulates natural selection through a computational fitness function which depends on the problem being addressed. These approaches require being able to build representations and genetic operators that are well suited for a given problen. More fundamentally perhaps these methods suffer also from computational complexity limitations since having to simulate the slow evolution of large populations over a large number of generations is inherently a computationally intensive process.

The graphical model approach [Pearl, 1988, Lauritzen, 1996, Heckerman, 1998, Frey, 1998] is a probabilistic approach where random variables are associated with the nodes of a graph and where the connectivity of the graph is directly related to Markovian independence assumptions between the variables. With structured data, the graph typically consists of inputs nodes directly reflecting the structure (sequence, graph, etc) of the input data, hidden nodes associated with hidden dynamics and context propagation, and output nodes associated with, for instance, classification or regression tasks. Both directed (Bayesian networks) and undirected (random Markov fields) formalisms can be used, or even combined. Graphical models are parameterized by local conditional distributions of a node variable given its neighbor variables – for instance a node variable given its parent variables in the case of Bayesian networks. In order to process data of variable size and format, particular assumptions must be made. These come typically in the form of stationarity or translation-invariance assumptions in regularly structured graphs (e.g. dynamic Bayesian networks), such as linear chains, trees with bounded degree, and lattices. Major challenges in the graphical model approach are the choice of suitable graphs and random variables and the propagation of information and learning which, in complex models, can be computationally demanding. Graphical models, such as Hidden Markov Models, have been very successful in bioinformatics, for instance, in order to model biological sequences [Baldi and Brunak, 2001]. Their application to molecular structures, however, has been more limited in part because of the computational challenges. An example of application of graphical models to the prediction of protein side-chains can be found in Yanover and Weiss [2003].

The recursive neural network (RNN) approach [Baldi and Chauvin, 1996, Goller and Kuchler, 1996, Sperduti and Starita, 1997, LeCun et al., 1998, Frasconi et al., 1998, Micheli et al., 2001, 2003, Baldi and Pollastri, 2003] can be viewed as a variation of the graphical model approach with Bayesian networks. The key difference is that relationships between graph variables are deterministic and parameterized by neural networks. Stationarity assumptions, also known as weight-sharing, lead to recurrent (in time) or more generally to recursive (in space and time) neural networks. The directed acyclic nature

of the underlying graph allows error gradients to be back-propagated. The loss of semantic power and flexibility imposed by the deterministic relationships is compensated by a considerable increase in propagation and learning speeds, although gradient descent in RNNs requires often some delicate tuning. Note that while the guts of a RNN are deterministic, the overall model can remain probabilistic since, for instance, in a classification task the activity of the (normalized exponential) output units can be interpreted as class-membership probabilities. Indeed, RNNs can be viewed formally as a limiting case of Bayesian networks when the local density functions approach Kronecker or Dirac delta functions [Baldi and Rosen-Zvi, 2005]. The RNN approach has been successfully applied to problems in protein structure prediction [Pollastri et al., 2001, Baldi and Pollastri, 2003] and some problems in computational chemistry [Micheli et al., 2001, 2003]. The major challenges in the recursive neural network approach, shared with graphical models, are first the design of the underlying acyclic graph and then the choice of the structure and complexity of the neural networks used to parameterize the relationships between node variables. In particular, when RNNs are applied to molecular data represented by graphs of covalent bonds, important issues that need to be resolved are the acyclic orientation of the bonds and possibly the selection of a center for each molecule. These operations can be done but introduce some degree of arbitrariness due to the lack of a canonical solution.

Finally, in recent years, kernel methods have emerged as an important class of machine learning methods suitable for variable-size structured data [Cristianini and Shawe-Taylor, 2000, Schölkopf and Smola, 2002]. Given two input objects $\mathbf{u}$ and $\mathbf{v}$, such as two molecules, the basic idea behind kernel methods is, to construct a kernel $k(\mathbf{u}, \mathbf{v})$ which measures the similarity between $\mathbf{u}$ and $\mathbf{v}$. This kernel can also be viewed as an inner product of the form $k(\mathbf{u}, \mathbf{v}) = \langle \phi(\mathbf{u}), \phi(\mathbf{v}) \rangle$ in an embedding feature space determined by the map ϕ which needs not be given explicitly. Regression, classification, and other tasks can then be tackled using linear (convex) methods based solely on inner products computed via the kernel in the embedding space, rather than the original input space. The challenge for kernel methods is to build or learn suitable kernels for a given task. Indeed, neural networks can be viewed as a special kind of kernel method where kernel parameters (associated with the feature embedding implemented by the lower layers) and linear weights (implemented by the output layer) are learnt simultaneously. Applications of kernel methods to graphical objects such as molecular bond graphs require the construction of graph kernels, that is functions that are capable of measuring similarity between graphs with labeled nodes and edges.

In the next sections, we first review kernel methods and some of the existing graph kernels, including graph kernels for chemical applications, that have been developed in the literature [Gärtner, 2003, Gärtner et al., 2003, Kashima et al., 2003]. We then develop new graph kernels for computational chem-

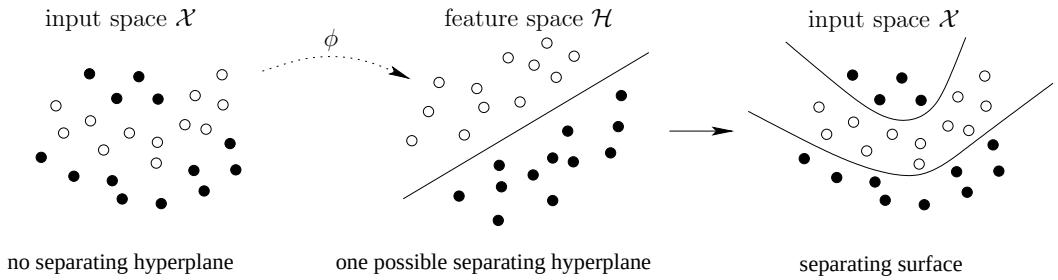

Fig. 1. The kernel approach for classification. Left: non-linearly separable input provided by black and white dots. Middle: perfect or approximate linear-separability can be achieved in *feature space* via the mapping ϕ. Right: linear decision surface in feature space defines a complex decision surface in input space.

istry applications based on molecular fingerprinting techniques and depth-first searches. We apply these kernels to the problem of predicting toxicity, mutagenicity, and cancer rescue activity on different data sets. Additional kernels and their tradeoffs are presented and discussed at the end.

2 Graph Kernels

2.1 Kernel Methods

Although the basic idea behind *kernel methods* is quite old (e.g. Kimeldorf and Wahba [1971]), over the last decade it has regained considerable interest spurred by the work of Boser et al. [1992] and Cortes and Vapnik [1995] on *support vector machines*. Many new kernel methods have been developed and/or successfully applied to challenging problems in recent years [Cristianini and Shawe-Taylor, 2000, Schölkopf and Smola, 2002].

In essence, kernel methods handle non-linear complex tasks using linear methods in a new space (Figure 1). To fix the ideas, consider a classification problem with training set $\mathcal{S} = \{(\mathbf{u}_1, y_1), \ldots, (\mathbf{u}_\ell, y_\ell)\}$, $(\mathbf{u}_i, y_i) \in \mathcal{X} \times \mathcal{Y}, i = 1, \ldots, \ell$, where $\mathcal{X}$ is an inner-product space (e.g. $\mathbb{R}^d$) with inner product denoted by $\langle \cdot, \cdot \rangle$, and $\mathcal{Y} = \{-1, +1\}$. In this setting, learning is the task of building a function $f \in \mathcal{Y}^{\mathcal{X}}$ from the training set $\mathcal{S}$ associating a *class* $y \in \mathcal{Y}$ to a *pattern* $\mathbf{u} \in \mathcal{X}$ such that the *generalization error* of f is as low as possible (for a more formal presentation of kernels and *statistical learning theory* see, for instance, [Vapnik, 1998, Herbrich, 2002, Schölkopf and Smola, 2002]).

A simple functional form for f is the hyperplane: $f(\mathbf{u}) = \text{sign}\,(\langle \mathbf{w}, \mathbf{u} \rangle + b)$, where $\text{sign}\,(\cdot)$ is the function returning the sign of its argument. The decision function f outputs a prediction depending on which side of the hyperplane $\langle \mathbf{w}, \mathbf{u} \rangle + b = 0$ the input pattern $\mathbf{u}$ lies. Under reasonable assumptions dis-

cussed in the references (e.g. maximum margin classification), solving for the "best" hyperplane leads to a convex quadratic optimization problem such that the solution vector $\mathbf{w}$ is a (usually sparse) linear combination of the training vectors: $\mathbf{w} = \sum_{i=1}^{\ell} \alpha_i y_i \mathbf{u}_i$, for some $\alpha_i \in \mathbb{R}^+, i = 1, \ldots, \ell$ (e.g. Müller et al. [2001]). More generally, this is known as the *representer theorem* [Kimeldorf and Wahba, 1971, Schölkopf et al., 2000]. Thus the linear classifier f can be rewritten as

$$f(\mathbf{u}) = \text{ sign } \left(\sum_{i=1}^{\ell} \alpha_i y_i \langle \mathbf{u}_i, \mathbf{u} \rangle + b \right) \tag{1}$$

However, for complex classification problems (Figure 1), the set of all possible linear decision surfaces may not be rich enough to provide good classification, no matter what the values of the parameters $\mathbf{w} \in \mathcal{X}$ and $b \in \mathbb{R}$ are. The purpose of the *kernel trick* [Aizerman et al., 1964, Boser et al., 1992], is precisely to overcome this limitation by applying a linear approach to the transformed data $\phi(\mathbf{u}_1), \ldots, \phi(\mathbf{u}_l)$ rather than the raw data. Here ϕ denotes an embedding function from the input space $\mathcal{X}$ to a feature space $\mathcal{H}$, equipped with a dot product. Using the representer theorem, the separating function must now be of the form:

$$f(\mathbf{u}) = \text{ sign } \left(\sum_{i=1}^{\ell} \alpha_i y_i \langle \phi(\mathbf{u}_i), \phi(\mathbf{u}) \rangle + b \right) \tag{2}$$

The key ingredient in the kernel approach is to replace the dot product in feature space with a kernel $k(\mathbf{u}, \mathbf{v}) = \langle \phi(\mathbf{u}), \phi(\mathbf{v}) \rangle$ using the definition of positive definite kernels, Gram matrices, and Mercer's theorem.

Definition 1 (Positive definite kernel) *Let $\mathcal{X}$ be a nonempty space. Let $k \in \mathbb{R}^{\mathcal{X} \times \mathcal{X}}$ be a continuous and symmetric function. k is a positive definite kernel iff for all $\ell \in \mathbb{N}$, for all $\mathbf{u}_1, \ldots, \mathbf{u}_\ell \in \mathbb{R}$, the square $\ell \times \ell$ matrix $K = (k(\mathbf{u}_i, \mathbf{u}_j))_{1 \leq i,j \leq \ell}$ is positive semi-definite, i.e. all its eigenvalues are nonnegative.*

For a given set $\mathcal{S}_u = \{\mathbf{u}_1, \ldots, \mathbf{u}_\ell\}$, K is called the Gram matrix of k with respect to $\mathcal{S}_u$. Positive definite kernels are also referred to as Mercer kernels.

Theorem 2 (Mercer's property) *For any positive definite kernel function $k \in \mathbb{R}^{\mathcal{X} \times \mathcal{X}}$, there exists a mapping $\phi \in \mathcal{H}^{\mathcal{X}}$ into the feature space $\mathcal{H}$ equipped with the inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$, such that:*

$$\forall \mathbf{u}, \mathbf{v} \in \mathcal{X} \quad k(\mathbf{u}, \mathbf{v}) = \langle \phi(\mathbf{u}), \phi(\mathbf{v}) \rangle_{\mathcal{H}}$$

The kernel approach consists in replacing all inner products in Equation 2 and all related expressions for computing the real coefficients α_i and b, by a

Mercer kernel k. For any input pattern $\mathbf{u}$, the corresponding decision function f is given by:

$$f(\mathbf{u}) = \text{ sign } \left(\sum_{i=1}^{\ell} \alpha_i y_i k(\mathbf{u}_i, \mathbf{u}) + b \right) \tag{3}$$

The kernel approach is equivalent to transforming the input patterns $\mathbf{u}_1, \ldots, \mathbf{u}_\ell$ into the corresponding vectors $\phi(\mathbf{u}_1), \ldots, \phi(\mathbf{u}_\ell) \in \mathcal{H}$ through the mapping $\phi \in \mathcal{H}^{\mathcal{X}}$ (cf. Mercer's property, Theorem 2), and to use hyperplanes in the feature space $\mathcal{H}$ for classification, as illustrated in Figure 1. While the natural dot product $\langle \mathbf{u}, \mathbf{v} \rangle = \sum_{i=1}^{d} u_i v_i$ of $\mathbb{R}^d$ is indeed a Mercer kernel, other well-known Mercer kernels, such as polynomial and gaussian kernels, usually correspond to nonlinear mappings ϕ into high-dimensional (even infinite-dimensional) feature spaces $\mathcal{H}$. The Gram matrix implicitly defines the geometry of the embedding space and allows the use of linear techniques in feature space to derive complex decision surfaces in input space $\mathcal{X}$. The same ideas can be applied beyond classification problems to, for instance, regression or unsupervised tasks (e.g. kernel PCA). In fact, any linear algorithm that relies exclusively on dot products between inputs can easily be "kernelized" [Friess et al., 1998, Mika et al., 1999, Müller et al., 2001, Bach and Jordan, 2002].

In summary, the application of kernel methods require two independent modules: (1) a module for computing the kernel and the Gram matrix; and (2) a module for computing the optimal manifold in feature space (e.g. a hyperplane in binary classification problems). Since the second module is usually readily available "off-the-shelf", the main effort for processing structured, variable-size data ought to go into the design of suitable kernel functions to assess similarity between input patterns. The design of efficient graph kernels in chemistry is an important step towards addressing the difficult problem – particularly in organic chemistry – of classifying compounds according to their physical, chemical, or biological properties. Thus we turn now to the design of efficient kernels for molecular structures represented by labeled, undirected, graph of bonds, with labels assigned to both nodes (atoms) and edges (bond type).

2.2 Convolution and Spectral Kernels

All the graph kernels discussed in this paper are special cases of, or related to, convolution/spectral kernels. Given two kernels k_1 and k_2 over the same set of objects, new kernels can be built using several operations, including convex linear combinations and convolutions. The convolution of k_1 and k_2 is a new kernel k with the form

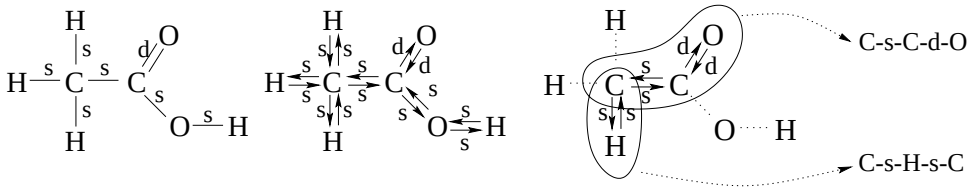

Fig. 2. Left: molecule represented as an undirected labeled graph. Vertex labels correspond to atom symbols and edge lables correspond to bond type (e.g. 's' for single bond, 'd' for double bond). Middle: graph orientation. Right: example of two paths that may be used by generic graph kernels. The path C-s-H-s-C defines a rather irrelevant feature, whereas the path path C-s-C-d-O is more informative.

$$k_1 \star k_2(\mathbf{u}, \mathbf{v}) = \sum_{(\mathbf{u}_1,\mathbf{u}_2)=\mathbf{u};(\mathbf{v}_1,\mathbf{v}_2)=\mathbf{v}} k_1(\mathbf{u}_1, \mathbf{v}_1) k_2(\mathbf{u}_2, \mathbf{v}_2) \tag{4}$$

where $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2)$ denotes a partition of $\mathbf{u}$ into two substructures $\mathbf{u}_1$ and $\mathbf{u}_2$ [Haussler, 1999, Schölkopf and Smola, 2002]. The nature of the substructures depends on the domain of course and could be, for instance, subsets or substrings in the case of kernels defined over sets or strings. Spectral kernels can be viewed as a special case of convolution kernels based on feature vectors derived by counting substructures contained in a given structure. In the case of graphs, the substructures are subgraphs. Different kernels can be derived by considering different classes of subgraphs (e.g. directed/undirected, labeled/unlabeled, paths/trees/cycles/subgraphs, deterministic/random walks, bounded/unbounded size) and different ways of listing and counting them (e.g. binary indicators/number of occurrences, depth-first/breadth-first). Due to the combinatorial explosion associated with variable-size substructures, space and time complexity consideration for computing convolution/spectral kernels are important.

Within the general framework of convolution kernels, two main approaches have been proposed for the design of graph kernels: (1) graph kernels based on powers of adjacency matrices [Gärtner, 2003, Gärtner et al., 2003]; and (2) marginalized graph kernels [Kashima et al., 2003]. For completeness, we review both approaches below. However, when applied to real-world chemical compound problems, these approaches are faced with several challenges such as the problem of orienting the unoriented bonds (Figure 2) and the computational cost involved in evaluating the kernel function. In addition, these kernels involve parameters that cannot be set easily, such as the parameters of the graph-product kernel of Gärtner et al. [2003].

2.3 Notation

An undirected graph $G = (\mathcal{V}, \mathcal{E})$ is defined by a finite set of vertices $\mathcal{V} = \{v_1, \ldots, v_n\}$ and a finite set of edges $\mathcal{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_m\}$. In the chemistry applications, we consider vertex- and edge-labeled graphs with two finite sets of labels $\mathcal{A} = \{l_1^a, \ldots, l_p^a\}$ for the atoms and $\mathcal{B} = \{l_1^b, \ldots, l_q^b\}$ for the bonds. The $m \times m$ adjacency matrix E of a graph G is such that its (i, j)-coefficient E_{ij} is equal to 1 if and only if there is an edge between vertices v_i and v_j. A *walk* $\mathbf{h} = [h_1, \ldots, h_{s+1}]$ of length s is a sequence of integers in the range $1, \ldots, n$ such that there exists an edge between each pair of vertices $(v_{h_i}, v_{h_{i+1}})$. The label $l(\mathbf{h})$ of a walk $\mathbf{h}$ is the string obtained by concatenating the labels of all the vertices and edges encountered along the walk. A *path* is a self avoiding walk, i.e. a walk that never goes through the same edge twice. A path, however, can visit a vertex more than once and therefore can contain cycles.

2.4 Graph Kernels Based on Powers of Adjacency Matrices

Here we describe two sub-classes of graph kernels, proposed in Gärtner et al. [2003], that are derived from the adjacency matrix and exploit label information to various degrees.

2.4.1 Kernels Based on Labeled Pairs

For a graph G, let the vertex-label matrix L be the $p \times n$ matrix such that its (r, i)-coefficient L_{ri} is equal to 1 if and only if the label of vertex i is equal to l_r^a.

The *labeled-pair graph kernels* described in Gärtner [2003] are defined by:

$$k(G_1, G_2) = \left\langle L_1 \left(\sum_{i=0}^{\infty} \lambda_i E_1^i \right) L_1', L_2 \left(\sum_{i=0}^{\infty} \lambda_i E_2^i \right) L_2' \right\rangle \tag{5}$$

where M' denotes the transpose of M. The inner product $\langle \cdot, \cdot \rangle$ is the Frobenius matrix product. The Frobenius product of two $m \times n$ matrices A and B is defined by the trace $\mathrm{tr}\,(A^* B)$, where $A^* = \bar{A}'$ is the conjugate transpose of A. These kernels are called labeled-pair kernels because, given two graphs G_1 and G_2, they count the number of walks in G_1 and G_2 of the same length and with the same labels on their first and last nodes. Within this family, three specific kernels have been proposed that can be specified in terms of the real coefficients λ_i and the sum $\sum_{i=0}^{\infty} \lambda_i E^i$ described below:

(1) exponential kernel:

$$\sum_{i=0}^{\infty} \lambda_i E^i = \sum_{i=0}^{\infty} \frac{(\gamma E)^i}{i!}$$

(2) truncated power series kernel:

$$\sum_{i=0}^{\infty} \lambda_i E^i = \sum_{i=0}^{p} (\gamma E)^i$$

(3) convergent geometric kernel:

$$\sum_{i=0}^{\infty} \lambda_i E^i = (1 - \gamma E)^{-1}$$

when the infinite power series is convergent ($p = \infty$).

The problem with kernels based on labeled pairs is that the feature space associated with such kernels is of dimension $|\mathcal{A}|^2$, i.e. the square of the number of different labels. Hence, if the number of labels for the atoms is low, the expressivity of the kernel is limited and may not be able to take into account important features of chemical compounds. Moreover, these kernels are derived from counting walks of the same length having the same start and end points, without giving any consideration to the sequence of nodes traversed by the walks. Clearly the actual sequence of atoms and bonds traversed along a walk often contains relevant information. Finally, these kernels give equal importance to uninformative (Figure 2) and possibly noisy and/or self-intersecting walks. To overcome some of these limitations a richer class of kernels is proposed in Gärtner et al. [2003]. These kernels rely also on adjacency matrices but use shared labeled sequences to compute graph similarity.

2.4.2 Kernels Based on Sequences of Labels (Product Graph Kernels)

To count shared labeled sequences, *product graph kernels* use an elegant, polynomial-time (with respect to the size of the graphs), procedure. This procedure is based on the direct graph product of two graphs [Gärtner et al., 2003].

Let $G_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $G_2 = (\mathcal{V}_2, \mathcal{E}_2)$ be two vertex- and edge-labeled graphs. The direct graph product $G_\times = (\mathcal{V}_\times, \mathcal{E}_\times)$ of G_1 and G_2 is defined as:

$$\begin{aligned}\mathcal{V}_\times &= \{(v_1, v_2) \in \mathcal{V}_1 \times \mathcal{V}_2 : (label(v_1) = label(v_2))\} \\ \mathcal{E}_\times &= \{((u_1, u_2), (v_1, v_2)) \in \mathcal{V}_\times \times \mathcal{V}_\times : (u_1, v_1) \in \mathcal{E}_1 \\ &\quad \wedge (u_2, v_2) \in \mathcal{E}_2 \wedge (label(u_1, v_1) = label(u_2, v_2))\}\end{aligned}$$

and a vertex (edge) in $G_\times$ is assigned the same label as the corresponding vertices (edges) in G_1 and G_2.

Given two graphs G_1 and G_2, the graph product kernel $k_\times$ is defined by:

$$k_\times(G_1, G_2) = \sum_{i,j=1}^{|\mathcal{V}_\times|} \left[\sum_{n=0}^{\infty} \lambda_n E_\times^n\right]_{ij} \tag{6}$$

for a suitable choice of $\lambda_0, \lambda_1, \lambda_2, \ldots$. This kernel corresponds to a weighted sum of the number of shared labeled sequences in the graphs G_1 and G_2. Note that the exponential, truncated power series, and convergent geometric kernels described in the previous sections are special cases of product graph kernels. For two graphs $G1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $G2 = (\mathcal{V}_2, \mathcal{E}_2)$, the product graph has $O(|\mathcal{V}_1||\mathcal{V}_2|)$ vertices and $O(|\mathcal{E}_1||\mathcal{E}_2|)$ edges. To compute the product graph kernel requires $O(|\mathcal{V}_\times|^3)$ steps since it is sufficient to compute the eigenvalues of $E_\times$, which takes $O(|\mathcal{V}_\times|^3)$ steps, to efficiently evaluate its powers.

2.5 Marginalized Graph Kernels

Marginalized graph kernels are another class of related graph kernels derived as a a generalization of *marginalized sequence kernels* [Tsuda et al., 2002]. Here, again, the similarity between two graphs is computed with respect to the number of labeled walks they share. Unlike the case of product graph kernels, marginalized kernels use Markov *random* walks on the underlying graph. First-order Markov walks are described by a transition probability matrix p_{ij}, where p_{ij} is the probability of moving from vertex v_i to vertex v_j, and the initial probability vector p_i [Kashima et al., 2003].

The general form for these kernels is:

$$k_m(G_1, G_2) = \sum_{i=1}^{\infty} \sum_{\substack{\mathbf{h}^1=[h_1^1,\ldots,h_i^1] \\ \mathbf{h}^2=[h_1^2,\ldots,h_i^2]}} k_{label}(l_1(\mathbf{h}^1), l_2(\mathbf{h}^2)) p(\mathbf{h}^1|G_1) p(\mathbf{h}^2|G_2)$$

where $l_g(\mathbf{h}^g)$ is the label of random walk $\mathbf{h}^g$ with respect to graph G_g, k_{label} is a specific kernel on labels (i.e. on strings), and $p(\mathbf{h}^g|G_g)$ is the probability of generating walk $\mathbf{h}^g$ from G_g given the transition parameters p_{ij} and the initial probabilities p_i.

More recently, there have been attempts to combine random walks and graph products [Mahé et al., 2004]. To computer these kernels, Kashima et al. [2003] propose a two-step approach consisting of an iterative procedure followed by the solution of a corresponding system of linear equations. In addition, these authors derive a sufficient condition on k_{label} to ensure finiteness of the marginalized kernel value. Marginalized kernels have been further enhanced to reduce computational costs by increasing the specificity of the vertex labels and thus reducing the number of common walks between two graphs. In

addition, the Markov process driving the generation of random walks [Mahé et al., 2004] can be modified in order to remove "totters", i.e. irrelevant loops resulting from the orientation of the bonds. Irrelvent paths, such as C-H-C in Figure 2 can be avoided by, for instance, using a second-order Markov chain that assigns 0 probability to such short loops. In spite of these improvements, the computation of these kernels requires an iterative procedure that remains computationally expensive. Furthermore, higher-order Markov random walk models require more complex sets of parameters that need to be determined.

2.6 Other Graph Kernels

There are several other classes of graphs kernels, such as diffusion kernels [Kondor and Lafferty, 2002]. Diffusion kernels, however, do not consider graph instances, but rather instances that are vertices of a space whose structure is described by a graph. Thus in general the problems that can be tackled with diffusion kernels are different from those considered here (see, for example, Vert and Kanehisa [2003]).

Tree kernels [Collins and Duffy, 2002, Vert, 2002, Vishwanathan and Smola, 2003] have also been studied in the literature. The central idea used to define this class of kernels is to count shared subtrees between two graphs. String kernels [Leslie et al., 2003, Lodhi et al., 2000] based on noncontiguous substrings can also be viewed as a particular case of tree kernels [Vishwanathan and Smola, 2003].

Recently, Horváth et al. [2004] have proposed *cyclic pattern kernels*, derived by counting common occurrences of *cycles* and *trees* between two graphs. Although they have demonstrated the applicability of cyclic pattern kernels on a chemical classification problem, these kernels do not take directly into account the important information provided by simple walks/paths.

Finally, Fisher kernels [Jaakkola and Haussler, 1998] use a probabilistic generative model of the training objects to build a fixed-length feature vector based on the derivatives of the likelihood with respect to the parameters of the generative model. The Fisher matrix, or some approximation thereof, is then used to compute the kernel. While the idea of Fisher kernels is very general, deriving suitable probabilistic generative models for graphs in general, and molecular compounds in particular, remains a challenging task. It is clear, however, that simple generative tree models could be tested.

3 Convolution Kernels for Chemistry

In this section, we develop classes of graph kernels that can be viewed as spectral kernels derived by counting and comparing walks in the underlying graphs. To circumvent the limitations posed by other graph kernels and address problems of molecular classification, we propose to use and extend the common technique of *molecular fingerprinting* [Flower, 1998, Raymond and Willett, 2001] used by chemists to build new feature vector representations of molecules. What differentiates our approach from the kernels in the previous section are the types of feature vectors we derive and how similarity between feature vectors is computed. The kernels we derive can be computed efficiently and leverage the peculiar properties of small-molecule graphs in organic chemistry. In particular these graphs are fairly small both in terms of the number of vertices and the number of edges and they are highly constrained by the laws of chemistry. These graphs have at most a few dozen vertices and the average degree is typically only slightly above two.

3.1 Molecular Fingerprinting

Traditional fingerprints are bit-vectors of a given size l, typically taken in the range of 100-1,000 (usually $l = 512$ or 1024). Given a molecule M having n atoms and m bonds, building a corresponding fingerprint requires starting *depth-first search* explorations from each atom in the molecule. Thus the substructures being considered are labeled paths, which may include labeled cycles. A hash value v is computed for each path described by the sequence of atoms and bonds visited (e.g. C-s-C-d-O, see Figure 2). For each such path, v is used to initialize a random number generator and b integers are produced (typically $b = 1$ or $b = 4$). The b integers are reduced modulo l and the corresponding bits are set to one in the fingerprint. If a bit in the nascent fingerprint is set to one by a path, it is left unchanged by all the other paths (i.e. "$1 + 1 = 1$" in case of a clash).

A very attractive feature of these bit vectors is that if the maximal path length d is set to $+\infty$, i.e. if we want to extract all the paths starting from all the atoms of a molecule, the complexity of the procedure is only $O(nm)$ if we do not allow path emanating from a vertex to share edges once they have diverged (see cases 1 and 2 in Section 3.2). In practice, d is often set to a lower value, typically in the range of eight to ten. Moreover, since fingerprinting is commonly used by chemists, typical values for l, b, and d are readily available together, if needed, with additional information, such as irrelevant paths that can be discarded. Lastly, a growing number of chemical databases already include some kind of fingerprint field in their tables, although standards have

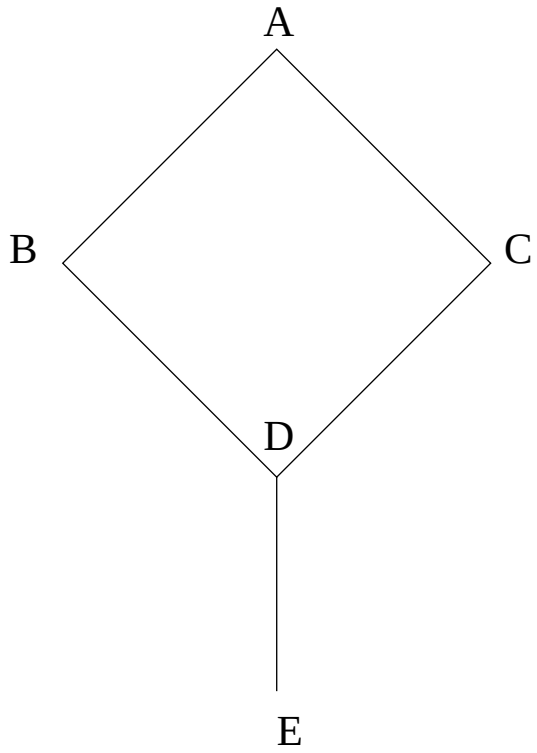

Fig. 3. Example of simple graph containing a loop used to illustrate various options in the depth first search extraction of paths of depth $d = 4$ (see text).

not yet emerged.

3.2 Depth-First Search Implementation

In a tree, starting from a fixed node A, depth first search exploration of depth d yields a list of all the paths of length d emanating from that node. However, because molecular graphs contain cycles, there can be different implementations of the depth-first seach leading to different sets of paths. Variations arise depending on whether cycles are allowed or not in a given path and whether two different paths, emanating from the same node, are allowed to share any edges after the first point of divergence from each other. The four possible resulting variations are illustrated using the graph of Figure 3 starting from node A with the ordering induced by alphabetic ordering and depth up to $d = 4$.

(1) DFS with no cycles:
- A
- A-B
- A-B-D
- A-B-D-C
- A-B-D-E

- A-C
- Note that A-B-D-C-A is not allowed because it contains a cycle, and A-C-D is not allowed because C-D has already been traversed.

(2) DFS with cycles:
- A
- A-B
- A-B-D
- A-B-D-C
- A-B-D-C-A (cycles are OK)
- A-B-D-E
- Note that A-C is not allowed because A-C has already been traversed.

(3) DFS with all paths and no cycles:
- A
- A-B
- A-B-D
- A-B-D-C
- A-B-D-E
- A-C
- A-C-D (traversing CD twice is OK)
- A-C-D-B
- A-C-D-E
- Note that A-B-D-C-A is not allowed because it contains a cycle.

(4) DFS with all paths and with cycles:
- A
- A-B
- A-B-D
- A-B-D-C
- A-B-D-C-A (cycles are OK)
- A-B-D-E
- A-C (traversing A-C twice is OK)
- A-C-D
- A-C-D-B
- A-C-D-B-A (cycles are OK)
- A-C-D-E

For a molecule with n atoms and m edges, the complexity of extracting all the paths (of any depth) is $O(nm)$ in the first and the second cases. In the third and fourth cases, for paths of length up to d, the complexity is at most $O(n\alpha^d)$, where α is the branching factor. Because the average degree of graphs in organic chemistry is typically only slightly above two, this branching factor is only slightly above 1. This still yields very reasonable time complexity estimates for typical values of d, such as $d = 10$. The implementation that does not allow sharing of edges once two paths have diverged is faster, but also less balanced because the resulting paths around cycles depend on the ordering of the nodes and edges used to define the depth-first search.

3.3 Generalized Fingerprints

In addition to conventional fingerprints, we use several generalized fingerprints. To circumvent the loss of information associated with clashes, for example, we use long binary feature vectors with a unique bit position reserved for each possible path. We also consider fingerprints constructed using path counts rather than binary indicator variables. We also consider weighting the vectors according to the TF-IDF scheme [Salton, 1991] commonly used in text retrieval. In this case, a molecule can be viewed as a text document consisting of all the labeled paths of length up to d that can be retrieved by depth-first searches. Our preliminary experiments using the TF-IDF approach, however, did not yield any significant improvements and therefore it is not used here. An alternative to the TF-IDF scheme, originated also in the field of information retrieval, is to consider a reduced set of paths selected according to the mutual information criterion [Yang and Pedersen, 1997, Dumais et al., 1998] in order to preserve or enhance paths carrying the most relevant information for a given classification task. An interesting aspect of this approach, besides that of reducing the path vocabulary size, is that the automatically-extracted paths may be validated (or invalidated) by chemists.

To summarize the situation, let $\mathcal{P}(d)$ be the set of all possible atom-bond labeled paths containing a maximum of d bonds. Using a depth-first search approach, the binary *feature map* $\boldsymbol{\phi}$ for a molecule $\mathbf{u}$ and a given depth d can be written as:

$$\boldsymbol{\phi}_d(\mathbf{u}) = (\phi_{path}(\mathbf{u}))_{path \in \mathcal{P}(d)}$$

Here $\phi_{path}(\mathbf{u})$ is equal to 1 if at least one depth-first search of depth less or equal to d starting from one of the atoms of $\mathbf{u}$ produces the path *path*. The counting feature map $\boldsymbol{\varphi}_d$ is defined similarly by using φ_{path} to count the number of labeled paths of each kind. The feature map giving fixed-size vectors of size l corresponds to the particular feature map $\bar{\boldsymbol{\phi}}_{d,l}$ given by:

$$\bar{\boldsymbol{\phi}}_{d,l}(\mathbf{u}) = (\phi_{\gamma_l(path)}(\mathbf{u}))_{path \in \mathcal{P}(d)}$$

where $\gamma_l : \mathcal{P}(d) \to \{1, \ldots, l\}^b$ is a function mapping paths to (set of) indices. Standard chemical fingerprints are a special case where the hash function, random generation, and congruence operations are captured by the function $\phi_{\gamma_l(path)}$.

3.4 Chemical Kernels Using Different Notions of Fingerprint Similarity

With these feature maps or generalized fingerprints, different kernels can be derived by using different measures of similarity between fingerprints. For two molecules $\mathbf{u}, \mathbf{v}$, the simple inner product $k_d(\cdot, \cdot) = \langle \cdot, \cdot \rangle_d$ yields the kernel:

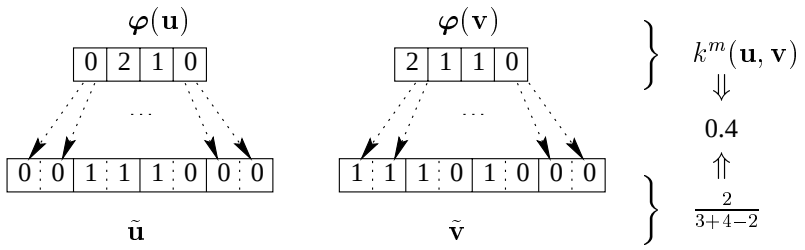

Fig. 4. Connection between Tanimoto and MinMax kernels. If the feature vectors $\boldsymbol{\varphi}(\mathbf{u})$ and $\boldsymbol{\varphi}(\mathbf{v})$ are transformed into the bit vectors $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ then $k_d^m(\mathbf{u}, \mathbf{v})$ is exactly the ratio between the number of bits set to one in both $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ divided by the total number of (unique) bits set to one in $\tilde{\mathbf{u}}$ or $\tilde{\mathbf{v}}$. Given a large integer q, each path in $\mathcal{P}(d)$ is associated with a distinct set of q consecutive indices. If $\varphi_{path}(\mathbf{u}) > 0$ then $\varphi_{path}(\mathbf{u})$ consecutive bits are set to one in $\tilde{\mathbf{u}}$ starting from the indices corresponding to *path*. The same holds for $\mathbf{v}$ and $\tilde{\mathbf{v}}$.

$$k_d(\mathbf{u}, \mathbf{v}) = \langle \mathbf{u}, \mathbf{v} \rangle_d = \sum_{path \in \mathcal{P}(d)} \phi_{path}(\mathbf{u}) \phi_{path}(\mathbf{v})$$

The corresponding kernel $\bar{k}_{d,l}$ is defined in the same way, using $\bar{\boldsymbol{\phi}}_{d,l}$. Here we develop three normalized kernels closely related to the Tanimoto similarity measure in the chemistry literature [Fligner et al., 2002, Flower, 1998, Gower, 1971, Gower and Legendre, 1986].

Definition 3 (Tanimoto kernel) *Let* $\mathbf{u}, \mathbf{v}$ *denote two molecules and* d *be an integer. Consider the feature map* $\boldsymbol{\phi}_d$ *and the corresponding kernel* k_d*. The Tanimoto kernel* k_d^t *is defined by:*

$$k_d^t(\mathbf{u}, \mathbf{v}) = \frac{k_d(\mathbf{u}, \mathbf{v})}{k_d(\mathbf{u}, \mathbf{u}) + k_d(\mathbf{v}, \mathbf{v}) - k_d(\mathbf{u}, \mathbf{v})} \tag{7}$$

If $\boldsymbol{\phi}(\mathbf{u})$ is regarded as the set of features than can be extracted from $\mathbf{u}$ using depth-first search exploration, then k_d^t simply computes the ratio between $|\boldsymbol{\phi}(\mathbf{u}) \cap \boldsymbol{\phi}(\mathbf{v})|$, i.e. the number of elements in the intersection of the two sets $\boldsymbol{\phi}(\mathbf{u})$ and $\boldsymbol{\phi}(\mathbf{v})$, and $|\boldsymbol{\phi}(\mathbf{u}) \cup \boldsymbol{\phi}(\mathbf{v})|$, i.e. the number of elements in the set corresponding to the union of $\boldsymbol{\phi}(\mathbf{u})$ and $\boldsymbol{\phi}(\mathbf{v})$. If the feature map used is $\bar{\boldsymbol{\phi}}_{d,l}$ instead of $\boldsymbol{\phi}_d$, for a given $l \in \mathbb{N}$, and thus $\bar{k}_{d,l}$ instead of k_d, the corresponding kernel $\bar{k}_{dl}^t$ is exactly the Tanimoto similarity measure used in chemoinformatics for fast molecular comparison and retrieval with fixed-size bit vectors of size l.

Definition 4 (MinMax kernel) *Let* $\mathbf{u}, \mathbf{v}$ *denote two molecules and* d *be an integer. Consider the feature map* $\boldsymbol{\varphi}_d(\cdot)$*, and the corresponding* $\varphi_{path}(\cdot)$*. The*

MinMax kernel k_d^m is defined by:

$$k_d^m(\mathbf{u},\mathbf{v}) = \frac{\sum\limits_{path\in\mathcal{P}(d)} \min(\varphi_{path}(\mathbf{u}),\varphi_{path}(\mathbf{v}))}{\sum\limits_{path\in\mathcal{P}(d)} \max(\varphi_{path}(\mathbf{u}),\varphi_{path}(\mathbf{v}))} \tag{8}$$

This kernel function is closely related to the Tanimoto kernel in at least two different ways. First, it is identical to the Tanimoto kernel when applied to binary vectors. Second, in a more subtle way, the MinMax kernel can be viewed as a Tanimoto kernel on a different set of binary vectors obtained by transforming the vector of counts. More precisely, for a given d, consider the two integer-valued feature vectors $\boldsymbol{\varphi}_d(\mathbf{u})$ and $\boldsymbol{\varphi}_d(\mathbf{v})$ and an integer q larger than any count in $\boldsymbol{\varphi}_d(\mathbf{u})$ and $\boldsymbol{\varphi}_d(\mathbf{v})$. Expand $\boldsymbol{\varphi}_d(\mathbf{u})$ and $\boldsymbol{\varphi}_d(\mathbf{v})$ into two binary feature vectors $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ of size pq, with $p = |\mathcal{P}(d)|$, such that $\tilde{u}_i$ (resp. $\tilde{v}_{i'}$) is set to one if and only if $i \mod q < \tilde{u}_i$ (resp. $i' \mod q < \tilde{v}_{i'}$), This yields (Figure 4):

$$k_d^m(\mathbf{u},\mathbf{v}) = \frac{\langle\tilde{\mathbf{u}},\tilde{\mathbf{v}}\rangle}{\langle\tilde{\mathbf{u}},\tilde{\mathbf{u}}\rangle + \langle\tilde{\mathbf{v}},\tilde{\mathbf{v}}\rangle - \langle\tilde{\mathbf{u}},\tilde{\mathbf{v}}\rangle}$$

The MinMax kernel takes into account the frequency of different paths in a molecule and, like the Tanimoto kernel, its value is always between 0 and 1. Using path counts rather than binary indicator variables, the MinMax kernel produces a more reliable way of assessing similarity between molecules of different sizes (see Section 5).

Definition 5 (Hybrid kernel) *Let* $\mathbf{u}$ *and* $\mathbf{v}$ *denote two molecules and let* d *and* l *be two integers. Let* θ *be a real number in* $[-1,+2]$*. The Hybrid kernel* k_d^h *between* $\mathbf{u}$ *and* $\mathbf{v}$ *is defined by:*

$$k_{d,l}^h(\mathbf{u},\mathbf{v}) = \frac{1}{3}\Big((2-\theta)\cdot\bar{k}_{d,l}^t(\mathbf{u},\mathbf{v}) + (1+\theta)\cdot\neg\bar{k}_{d,l}^t(\mathbf{u},\mathbf{v})\Big) \tag{9}$$

where $\neg\bar{k}_{d,l}^t$ *is the 'Tanimoto' kernel based on the feature map* $(\neg\bar{\boldsymbol{\phi}}_{d,l}(\mathbf{x})) = (\neg\bar{\phi}_{\gamma_l(path)}(\mathbf{x}))_{path\in\mathcal{P}(d)}$*,* $\neg$ *is the logical negation, and* $\gamma : \mathcal{P}(d) \to 1,\ldots,l$*.*

Thus the Hybrid kernel is a convex combination of two kernels, respectively measuring the number of common paths and common missing-paths between two molecules [Fligner et al., 2002]. When $\theta = -1$, the Hybrid kernel reduces to the Tanimoto kernel. In practice, θ is typically set to the average density of the bit vectors, between 0 and 1. It should be clear that a hybrid version of the MinMax kernel is possible along the same lines. In simulations, however, the hybrid kernel did not yield any significant improvements and therefore it is only briefly mentioned in the simulation results. It should be noted that the kernels above are also conceptually related to the notions of precision/recall/sensitivity/specificity/F measures in classification, with the caveat that molecules, unlike positive and negative classes, play a symmet-

ric role. Indeed, the Tanimoto coefficient is essentially a variation on the F measure of information retrieval. The corresponding F kernel is given by:

$$k_d^t(\mathbf{u}, \mathbf{v}) = \frac{2k_d(\mathbf{u}, \mathbf{v})}{k_d(\mathbf{u}, \mathbf{u}) + k_d(\mathbf{v}, \mathbf{v})} \tag{10}$$

The F similarity is also a special case of the more general Tversky similarity [Tversky, 1977, Rouvray, 1992] defined by:

$$k_d^t(\mathbf{u}, \mathbf{v}) = \frac{2k_d(\mathbf{u}, \mathbf{v})}{\alpha k_d(\mathbf{u}, \mathbf{u}) + \beta k_d(\mathbf{v}, \mathbf{v})} \tag{11}$$

Proposition 6 *The Tanimoto kernel, MinMax kernel and Hybrid kernel are Mercer kernels.*

PROOF. A sketch of the proof that k_d^t, k_d^m, $k_{d,l}^h$ are Mercer kernels follows from a result given by Gower [1971] showing that, for any integer p and any set of ℓ binary vectors $\mathbf{x}_1, \ldots, \mathbf{x}_\ell \in \mathbb{R}^p$, the similarity matrix $S = (k^t(\mathbf{x}_i, \mathbf{x}_j)_{1 \leq i,j \leq \ell})$ is positive semi-definite. Thus k^t is a positive definite kernel or Mercer kernel. Given that for any Mercer kernel $k \in \mathbb{R}^{\mathcal{X} \times \mathcal{X}}$ and any mapping $g \in \mathcal{X}^{\mathcal{X}'}$, $k(g(\cdot), g(\cdot)) \in \mathbb{R}^{\mathcal{X}' \times \mathcal{X}'}$ is a Mercer kernel (see, for instance Schölkopf and Smola [2002]), the MinMax kernel is also a Mercer kernel. Finally, using the same argument, and the fact that a convex combination of Mercer kernels is a Mercer kernel, it follows that the Hybrid kernel is also a Mercer kernel.

In the following section we will use these kernels, with different parameter settings, to tackle three different chemical classification problems.

3.5 Fast Computation of Chemical Kernels

An important issue is whether these kernels can be computed efficiently. At first glance, the computations may appear prohibitive as the feature vectors produced by the feature map are of large dimension. However, using a *suffix tree* data structure [Ukkonen, 1995, Weiner, 1973] as also proposed in [Leslie et al., 2002, Vishwanathan and Smola, 2003], allows us to compute each of the proposed kernels in time $O(d(n_1 m_1 + n_2 m_2))$, where d is the depth of the search and n_i (resp. m_i) the number of vertices/atoms (resp. edges/bonds) of the two molecules considered. This time complexity is for the case where depth-first paths starting from the same vertex are not allowed to share edges once they have diverged.

More precisely, given a molecule M_1 consisting of n_1 atoms and m_1 bonds, performing all depth-first search explorations up to depth d starting from every atom is a process with complexity $O(n_1 m_1)$. This process produces $O(n_1 m_1)$ paths of length d or less. It is possible to store all those paths in a suffix tree resorting to an algorithm in $O(dn_1 m_1)$ [Ukkonen, 1995]. It is worth noting that in our implementation we do not distinguish the orientation of a given path (e.g. C-C-H is the same path as H-C-C) and the reverse of each path is also stored in the suffix tree. Following the same line of reasoning with a second molecule M_2 with n_2 atoms and m_2 bonds, and knowing that searching for a string s of length $|s|$ in a suffix tree takes $O(|s|)$ steps, $O(dn_2 m_2)$ steps are required to count the number of depth-first search paths extracted from M_2 that are also in the set of depth-first search paths found in M_1. The same holds conversely, exchanging the roles of M_1 and M_2. Thus it takes $O(d(n_1 m_1 + n_2 m_2))$ steps to compute the Tanimoto, MinMax, and Hybrid Kernels. This analysis applies to the implementation of depth-first search where paths emanating from a vertex are not allowed to share eges once they have diverged. If paths are allowed to share edges once they have diverged, the complexity may increase slightly but remains manageable, as discussed in Section 3.2. In short, these kernels can be computed very rapidly and hence can be used to tackle large-scale chemical classification and regression problems.

3.6 Voted Perceptron Classifier

Finally, in the simulations below we combine kernels with the Voted Perceptron learning algorithm. The Voted Perceptron algorithm, proposed by Freund and Schapire [1999], is an efficient learning algorithm based on the idea of voting perceptrons. It builds a classifier f of the form $f(\mathbf{x}) = \sum \alpha_t h_t(\mathbf{x})$ where each h_t is a perceptron, and makes its prediction according to the sign of $f(\mathbf{x})$. Freund and Schapire [1999] proposed a natural extension of this algorithm to the case where kernel perceptrons functions h_t are used instead of linear perceptrons to further enhance the quality of the classifier. The learning algorithm, summarized in pseudocode form below, is an iterative procedure which adds one perceptron at a time. While a specific maximum number of iterations can be fixed beforehand, running the algorithm until the training error stabilizes yields high-quality classifiers and avoids having to set additional hyperparameters.

Algorithm 1 (Voted Perceptron [Freund and Schapire, 1999])
input: $\mathcal{S} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$ *and a Mercer kernel* k
output: a list of weighted perceptrons $(\boldsymbol{\alpha}_1, c_1), \ldots, (\boldsymbol{\alpha}_k, c_k)$

- $p \leftarrow 0, \boldsymbol{\alpha}_1 \leftarrow \mathbf{0}, c_1 \leftarrow 0$
- **repeat** *until convergence of the training error*

· **for** $i = 1, \ldots, \ell$

compute prediction $\hat{y} \leftarrow sign\left(\sum_{j=1}^{\ell} \alpha_{pj} k(\mathbf{x}_j, \mathbf{x}_i)\right)$

if $\hat{y} = y_i$

$c_p \leftarrow c_p + 1$

else

$\boldsymbol{\alpha}_{p+1} \leftarrow \boldsymbol{\alpha}_p$

$\alpha_{p+1,i} \leftarrow \alpha_{pi} + y_i$

$c_{p+1} \leftarrow 1$

$p \leftarrow p + 1$

end if

· **end for**

• **end repeat**

4 Data

We apply the kernels introduced in the previous section together with the Voted Perceptron classifier to the problems of predicting mutagenicity, toxicity, and anti-cancer activity on three different data sets (Mutag, PTC, and NCI).

4.1 Mutag Dataset

The Mutag dataset [Debnath et al., 1991] consists originally of 230 chemical compounds assayed for mutagenicity in *Salmonella typhimurium* (Table 1). Among the 230 compounds, however, only 188 (125 positive, 63 negative) are considered to be learnable [Debnath et al., 1991] and thus are used in our simulations (Table 3). The results from other groups that are reported for comparison purposes were obtained also on the same subset of 188 molecules. The accuracy reported in the simulations is estimated by a leave-one-out procedure.

4.2 PTC Dataset

The Predictive Toxicology Challenge (PTC) dataset [Helma et al., 2001] reports the carcinogenicity of several hundred chemical compounds for Male Mice (MM), Female Mice (FM), Male Rats (MR) and Female Rats (FR)

(Table 1). As with the Mutag dataset, the accuracy reported in Table 3 is estimated by a leave-one-out procedure.

4.3 NCI Dataset

The Mutag and PTC datasets are useful but somewhat small. The NCI dataset, made publicly available by the National Cancer Institute (NCI), provides screening results for the ability of roughly 70,000 compounds to kill or inhibit the growth of a panel of 60 human tumor cell lines. We use the dataset corresponding to the concentration parameter GI50, essentially the concentration that causes 50% growth inhibition [1] . For each cell line, approximately 3,500 compounds, described by their 2D structures, are provided with information on their anti-tumor activity. The distributions of positive (compounds with anti-tumor activity) and negative examples for the 60 cell lines are reported in Table 2. Not only is the NCI dataset considerably larger than the Mutag and PTC datasets, but overall it is also more balanced. Thus the trivial background statistical predictor always predicting the class encountered more frequently has poorer performance on the NCI dataset. Performance on the NCI dataset is analyzed by cross validation methods using 20 random 80/20 training/test splits of each subset associated with each cell line. Values reported correspond to averages obtained over these 20 splits.

[1] A complete description of the cell lines is available at the url http://dtp.nci.nih.gov/docs/misc/common_files/cell_list.html

	Mutag	**MM**	**FM**	**MR**	**FR**
#pos.	125 (66.5%)	129 (38.4%)	143 (41.0%)	152 (44.2%)	121 (34.5%)
#neg.	63 (33.5%)	207 (61.6%)	206 (59.0%)	192 (55.8%)	230 (65.5%)
total ex.	188	336	349	344	351
Avg. #atoms/mol.	17.93	25.05	25.25	25.56	26.08
Avg. #bonds/mol.	19.79	25.39	25.62	25.96	26.53
Avg. degree	2.21	2.03	2.03	2.03	2.03

Table 1
Distribution of positive and negative examples and molecular graph statistics in the Mutag and PTC datasets.

Screen	#pos.	#neg.	Screen	#pos.	#neg.
786-0	1832 (52.3%)	1674 (47.7%)	NCI-H226	1781 (51.4%)	1683 (48.6%)
A498	1782 (51.2%)	1698 (48.8%)	NCI-H23	1968 (52.9%)	1751 (47.1%)
A549	1901 (50.9%)	1833 (49.1%)	NCI-H322M	1765 (47.8%)	1925 (52.2%)
ACHN	1795 (50.8%)	1736 (49.2%)	NCI-H460	2049 (56.9%)	1550 (43.1%)
BT-549	1399 (50.4%)	1379 (49.6%)	NCI-H522	2138 (59.8%)	1435 (40.2%)
CAKI-1	1865 (52.1%)	1715 (47.9%)	OVCAR-3	2001 (54.2%)	1690 (45.8%)
CCRF-CEM	2217 (63.7%)	1263 (36.3%)	OVCAR-4	1840 (51.4%)	1742 (48.6%)
COLO-205	1943 (53.3%)	1702 (46.7%)	OVCAR-5	1651 (45.0%)	2019 (55.0%)
DU-145	1416 (48.1%)	1529 (51.9%)	OVCAR-8	1979 (53.3%)	1735 (46.7%)
EKVX	1968 (53.5%)	1713 (46.5%)	PC-3	1522 (51.0%)	1460 (49.0%)
HCC-2998	1804 (56.8%)	1373 (43.2%)	RPMI-8226	2116 (59.4%)	1448 (40.6%)
HCT-116	2049 (55.0%)	1674 (45.0%)	RXF-393	1850 (54.4%)	1551 (45.6%)
HCT-15	1993 (53.4%)	1738 (46.6%)	SF-268	2020 (54.3%)	1701 (45.7%)
HL-60-TB	2188 (64.6%)	1198 (35.4%)	SF-295	2027 (54.1%)	1718 (45.9%)
HOP-62	1888 (52.0%)	1740 (48.0%)	SF-539	1920 (56.7%)	1464 (43.3%)
HOP-92	1982 (56.6%)	1521 (43.4%)	SK-MEL-28	1774 (47.6%)	1950 (52.4%)
HS-578T	1550 (54.0%)	1320 (46.0%)	SK-MEL-2	1783 (49.5%)	1817 (50.5%)
HT29	2004 (54.0%)	1708 (46.0%)	SK-MEL-5	2034 (55.2%)	1651 (44.8%)
IGROV1	1956 (53.0%)	1734 (47.0%)	SK-OV-3	1711 (48.8%)	1792 (51.2%)
K-562	2139 (59.1%)	1479 (40.9%)	SN12C	1918 (52.1%)	1764 (47.9%)
KM12	1941 (52.4%)	1764 (47.6%)	SNB-19	1840 (49.4%)	1885 (50.6%)
LOX-IMVI	2053 (57.0%)	1550 (43.0%)	SNB-75	2131 (61.1%)	1359 (38.9%)
M14	1815 (51.1%)	1736 (48.9%)	SR	1869 (62.2%)	1137 (37.8%)
MALME-3M	1886 (53.8%)	1621 (46.2%)	SW-620	1940 (51.7%)	1813 (48.3%)
MCF7	1733 (57.0%)	1306 (43.0%)	T-47D	1550 (53.3%)	1359 (46.7%)
MDA-MB-231	1475 (50.0%)	1473 (50.0%)	TK-10	1650 (47.3%)	1840 (52.7%)
MDA-MB-435	1519 (51.0%)	1462 (49.0%)	U251	2044 (54.4%)	1711 (45.6%)
MDA-N	1503 (50.7%)	1459 (49.3%)	UACC-257	1873 (50.9%)	1808 (49.1%)
MOLT-4	2175 (61.5%)	1359 (38.5%)	UACC-62	2046 (55.5%)	1638 (44.5%)
NCI-ADR-RES	1586 (51.0%)	1525 (49.0%)	UO-31	1994 (55.2%)	1621 (44.8%)

Table 2
Distribution of the 60 NCI screens studied. Positive examples correspond to anti-cancer activity.

5 Results

We conducted several experiments to compare the various classes of kernels with different parameter settings. Here we report representative subsets of results together with the main findings. In addition to the usual accuracy measure, we also measure the ROC score, which is the normalized area under the ROC curve plotting the proportion of *true positive* (TP) predictions as a function of the proportion of *false positive* (FP) predictions, as the classification threshold is varied [Hanley and McNeil, 1982]. Precision [TP/(TP+FP)] and recall [TP/(TP+FN)] measures are also computed together with their

harmonic mean (F-measure).

5.1 Mutag and PTC Datasets

The results on the Mutag and PTC datasets are reported in Table 3. In this table, we also report the results obtained by Kashima et al. [2003] using their marginalized kernels and a frequent pattern mining approach [Kramer and De Raedt, 2001]. A first observation that can be drawn from the table is that the Tanimoto kernel and the MinMax kernels for a depth $d = 10$ always rank among the top two methods and produce results that are significantly above chance, and consistently close or above those reported previously in the literature. In Table 3, we also see that using the 2D kernels in combination with the traditional molecular fingerprinting procedure, i.e. using fixed-size bit vectors of length l, yields good results. These results, however are not as good as those obtained with unbounded l and the deterioration in performance accentuates when l is reduced from 1024 to 512 (except for MM), most likely resulting from the increase in the number of clashes. On the Mutag dataset, the PD (Pattern Discovery) algorithm achieves 89.1% accuracy versus 87.8% for the Tanimoto kernel. Performances of up to 89.4% have actually been reported in King et al. [1996] (Table 4). Such a difference, however, may not be significant because the Mutag dataset is too small.

This is confirmed by further refining the 2D kernels and implementing an exhaustive search of *all* possible paths up to depth d, allowing paths to share edges once they have diverged, which can still be implemented efficiently as discussed in Section 3.2, thanks to the small size and small degree of these graphs (the average degree for the large NCI dataset, for instance, is 2.11). The corresponding results in Table 5 show, for example, that the MinMax kernel achieves a cross-validated performance accuracy of 91.5%, above all previously reported results [King et al., 1996, Mahé et al., 2004]. Table 5 shows also that the choice of atom labels can impact performance and that too fine-grained labels may not be optimal for datasets comparable in size to those used here.

Overall, these results indicate that the kernels introduced here, and particularly the 2D Tanimoto and MinMax kernel, are effective kernels for molecular classification tasks. We also conducted tests using a mutual information criterion [Yang and Pedersen, 1997, Dumais et al., 1998] to select informative paths. We computed the mutual information between the binary (0-1) variable associated with a given path and the binary (± 1) variable associated with the class. We ranked the paths accordingly and removed those corresponding to low mutual information. Surprisingly, for vectors of equal size, the results with mutual information are not better than those obtained using the simple, short-length, fingerprint approach. We conjecture that this is a size effect– the

Kernel/Method	**Mutag**	**MM**	**FM**	**MR**	**FR**
PD [Kashima et al., 2003]	**89.1**	61.0	61.0	62.8	66.7
MK [Kashima et al., 2003]	85.1	64.3	63.4	58.4	66.1
Tanimoto	*87.8*	**66.4**	*64.2*	63.7	*66.7*
MinMax	86.2	64.0	**64.5**	*64.5*	66.4
Tanimoto, $l = 1024, b = 1$	87.2	*66.1*	62.4	**65.7**	**66.9**
Hybrid $l = 1024, b = 1$	87.2	65.2	61.9	64.2	65.8
Tanimoto, $l = 512, b = 1$	84.6	**66.4**	59.9	59.9	66.1
Hybrid $l = 512, b = 1$	86.7	65.2	61.0	60.7	64.7
Tanimoto, $l = 1024$ + MI	84.6	63.1	63.0	61.9	*66.7*
Hybrid $l = 1024$ + MI	84.6	62.8	63.7	61.9	65.5
Tanimoto, $l = 512$ + MI	85.6	60.1	61.0	61.3	62.4
Hybrid $l = 512$ + MI	86.2	63.7	62.7	62.2	64.4

Table 3
Leave-one-out accuracy results for the Mutag and PTC datasets. b denotes the number of bits set to one for a given path and MI indicates that paths have been selected using the mutual information criterion. Depth of search is set to $d = 10$. The value of θ used for the Hybrid kernel is the average density of the fingerprints contained in the training set. Best results are in bold font and second best are italicized.

Lin. Reg.	**NNet**	**Dec. Trees**	**ILP**
89.3%	89.4%	88.3%	87.8%

Table 4
Accuracies for different machine learning methods on the Mutag problem reported in the literature: linear regression, neural networks, decision trees, inductive logic programming [King et al., 1996].

set of informative paths that is retained is too small–and a difference ougth to become noticeable with a larger value of l.

5.2 NCI dataset

In this larger set of experiments, we focused exclusively on the Tanimoto and the MinMax kernels since they gave the best preliminary results on the smaller Mutag and PTC datasets using the faster, but less exhaustive, implementation of depth-first search. The results for the 60 screens of the NCI dataset are

Kernel	**Type**	**Atom. #**	**Val.**
Tanimoto	87.8	90.4	90.4
Tanimoto+cycle	86.2	87.8	89.9
MinMax	89.4	91.0	**91.5**
MinMax+cycle	89.9	**91.5**	89.4

Table 5
Classification accuracies (%) obtained on Mutag using exhaustive path extraction and the Tanimoto and the MinMax kernel with a depth set to $d = 10$. 'Type' corresponds to the case where atom descriptions are fully retained (e.g. in the paths constructed, a carbon connected to two hydrogens is different from one connected to three), 'Atom. #' corresponds to the equivalence class where only the atomic numbers are used to label atoms and 'Val.' to the situation where all atoms having the same valence are considered equivalent.

reported in Tables 6 and 7 using a Tanimoto kernel of depth $d = 10$. Tables 8 and 9 show the results obtained using the MinMax kernel with depth $d = 10$. In each case, the first table reports the results in terms of accuracy and area under the ROC curve and the second table in terms of precision [TP/(TP+FP)] and recall [TP/(TP+FN)]. The mean accuracy for the Tanimoto kernel is 71.55% versus 72.29% for the MinMax kernel. Likewise, the mean ROC score is 77.86% versus 78.74%, the mean precision is 72.55% versus 73.02%, and the mean recall is 74.90% versus 76.05%.

On the NCI dataset, we observe in general that both kernels have performance accuracies above 70% in general, well above chance level for this balanced dataset. The MinMax kernel gives better results than the Tanimoto kernel almost consistently, albeit by a fairly small margin. We believe this results from the fact that this kernel uses actual counts rather than binary indicator variables. Using counts in the future ought to provide a richer representation and allow better comparison and classification of molecules.

6 Discussion

We have reviewed graph kernels and developed new graph kernels for chemical molecules that yield state-of-the art results on several datasets. These kernels measure the similarity between feature vectors, or molecular fingerprints, consisting of binary vectors or vectors of counts associated with all labeled paths of length less or equal to some value d derived by depth-first searches, starting from each vertex of a molecule. The accuracies obtained, for instance 72% on the NCI dataset, are encouraging and suggest that automatic classification of compounds may become a viable alternative, but not a substitute, to laborious

Screen	Accuracy (%)	ROC (%)	Screen	Accuracy (%)	ROC (%)
786-0	72.63 ± 1.33	78.78 ± 1.18	NCI-H226	70.17 ± 1.42	76.55 ± 1.55
A498	71.28 ± 2.08	77.82 ± 2.03	NCI-H23	71.64 ± 1.31	78.50 ± 1.22
A549	71.71 ± 1.29	78.77 ± 1.23	NCI-H322M	69.67 ± 1.62	76.00 ± 1.37
ACHN	72.59 ± 1.77	79.56 ± 1.79	NCI-H460	71.81 ± 1.63	77.82 ± 1.38
BT-549	70.63 ± 1.33	77.78 ± 1.29	NCI-H522	72.76 ± 1.51	78.48 ± 1.80
CAKI-1	72.20 ± 1.47	79.05 ± 1.29	OVCAR-3	71.81 ± 1.76	77.88 ± 1.59
CCRF-CEM	71.74 ± 1.62	75.95 ± 1.71	OVCAR-4	71.37 ± 1.41	77.95 ± 1.31
COLO-205	71.84 ± 1.68	78.53 ± 1.43	OVCAR-5	70.38 ± 1.53	77.04 ± 1.64
DU-145	71.79 ± 1.88	78.92 ± 1.85	OVCAR-8	71.97 ± 1.81	78.23 ± 1.32
EKVX	70.42 ± 2.11	76.41 ± 2.02	PC-3	72.50 ± 1.79	79.47 ± 1.73
HCC-2998	69.58 ± 1.44	75.39 ± 1.24	RPMI-8226	72.23 ± 1.30	77.57 ± 1.38
HCT-116	73.28 ± 1.63	79.75 ± 1.06	RXF-393	70.64 ± 1.33	77.26 ± 1.63
HCT-15	70.91 ± 1.23	77.55 ± 1.06	SF-268	72.35 ± 1.19	78.79 ± 1.84
HL-60-TB	72.04 ± 1.53	75.58 ± 1.88	SF-295	70.19 ± 1.35	77.16 ± 1.61
HOP-62	71.31 ± 1.37	77.89 ± 1.35	SF-539	70.95 ± 1.92	76.50 ± 1.63
HOP-92	70.30 ± 1.07	75.95 ± 1.15	SK-MEL-28	70.51 ± 1.31	77.25 ± 1.63
HS-578T	70.90 ± 1.54	76.89 ± 1.13	SK-MEL-2	71.54 ± 1.30	77.86 ± 1.39
HT29	71.89 ± 1.96	78.62 ± 2.10	SK-MEL-5	73.28 ± 1.30	79.76 ± 1.29
IGROV1	70.37 ± 1.62	76.79 ± 1.17	SK-OV-3	72.10 ± 1.80	78.70 ± 1.63
K-562	71.32 ± 1.55	76.63 ± 1.81	SN12C	70.77 ± 1.58	77.73 ± 1.31
KM12	70.82 ± 1.31	77.79 ± 1.68	SNB-19	71.44 ± 0.72	78.39 ± 1.04
LOX-IMVI	73.24 ± 1.36	79.39 ± 1.54	SNB-75	70.27 ± 1.37	74.66 ± 1.21
M14	72.10 ± 1.81	79.05 ± 1.69	SR	72.76 ± 1.47	77.28 ± 1.51
MALME-3M	71.14 ± 1.57	77.70 ± 1.43	SW-620	72.39 ± 1.34	79.42 ± 1.31
MCF7	73.59 ± 1.68	79.93 ± 1.57	T-47D	71.98 ± 1.77	79.25 ± 2.51
MDA-MB-231	71.21 ± 1.60	78.12 ± 1.28	TK-10	70.54 ± 1.86	76.70 ± 1.84
MDA-MB-435	71.98 ± 1.11	78.91 ± 1.35	U251	71.88 ± 1.42	78.49 ± 1.45
MDA-N	72.45 ± 1.20	79.14 ± 1.22	UACC-257	69.95 ± 1.58	76.71 ± 1.49
MOLT-4	72.89 ± 1.42	77.67 ± 1.31	UACC-62	72.57 ± 1.56	78.73 ± 1.67
NCI-ADR-RES	71.56 ± 1.66	78.28 ± 1.42	UO-31	70.79 ± 1.21	76.96 ± 1.56

Table 6
Classification accuracy and ROC score (and their standard deviations) obtained on the 60 NCI screens using k_{10}^{vt}

experimental characterization in the near future. Even if the accuracy is not perfect–leaving room for further improvements– batteries of such predictors could be used to sift through large set of molecules and rank them during screening experiments in drug discovery and other tasks.

One advantage of the feature vectors and kernels we have developed is that they can be computed very efficiently. The tradeoff, however, is that important information is sometimes discarded. This is apparent in our results where using binary vectors or vectors of fixed small length l degrades the performance with respect to using vectors of counts or large values of l. Likewise, we observe also a small degradation when depth-first search is implemented in a way that does

Screen	Precision (%)	Recall (%)	Screen	Precision (%)	Recall (%)
786-0	73.59 ± 2.80	74.90 ± 1.75	NCI-H226	70.84 ± 2.21	72.23 ± 2.48
A498	71.62 ± 3.54	72.52 ± 2.26	NCI-H23	72.62 ± 2.31	74.83 ± 1.97
A549	71.49 ± 2.82	72.76 ± 2.49	NCI-H322M	68.68 ± 2.45	67.51 ± 3.02
ACHN	72.03 ± 1.85	74.33 ± 2.69	NCI-H460	74.05 ± 2.43	77.20 ± 1.86
BT-549	71.72 ± 2.16	70.89 ± 2.80	NCI-H522	75.99 ± 2.18	79.74 ± 2.03
CAKI-1	72.55 ± 1.69	74.83 ± 2.72	OVCAR-3	73.35 ± 2.21	75.79 ± 2.27
CCRF-CEM	76.19 ± 2.14	81.15 ± 1.58	OVCAR-4	71.98 ± 1.68	72.84 ± 2.63
COLO-205	73.09 ± 2.16	74.98 ± 2.13	OVCAR-5	66.78 ± 1.80	65.77 ± 2.90
DU-145	70.65 ± 2.85	70.72 ± 2.44	OVCAR-8	73.20 ± 2.64	75.39 ± 2.56
EKVX	71.11 ± 2.26	75.21 ± 3.36	PC-3	72.28 ± 1.77	74.98 ± 3.15
HCC-2998	71.57 ± 1.90	76.39 ± 2.27	RPMI-8226	75.58 ± 1.44	79.49 ± 2.10
HCT-116	74.34 ± 2.23	77.59 ± 2.70	RXF-393	72.50 ± 1.79	74.14 ± 2.44
HCT-15	71.94 ± 1.70	74.66 ± 1.90	SF-268	73.90 ± 1.99	76.55 ± 2.30
HL-60-TB	76.31 ± 2.16	81.85 ± 2.07	SF-295	72.56 ± 1.99	73.14 ± 2.67
HOP-62	71.69 ± 2.19	73.56 ± 2.47	SF-539	73.56 ± 2.18	76.74 ± 3.05
HOP-92	72.26 ± 1.74	76.62 ± 2.01	SK-MEL-28	68.90 ± 2.42	67.73 ± 2.94
HS-578T	72.19 ± 2.33	75.21 ± 2.30	SK-MEL-2	71.31 ± 2.77	70.43 ± 2.25
HT29	73.59 ± 2.50	75.32 ± 2.37	SK-MEL-5	74.59 ± 1.42	77.97 ± 2.47
IGROV1	70.50 ± 1.84	73.74 ± 2.63	SK-OV-3	71.41 ± 2.92	71.37 ± 3.37
K-562	74.23 ± 1.82	78.96 ± 2.77	SN12C	71.61 ± 1.99	72.34 ± 2.87
KM12	71.36 ± 2.32	74.49 ± 2.36	SNB-19	71.26 ± 1.78	71.40 ± 1.92
LOX-IMVI	75.14 ± 1.51	79.49 ± 2.29	SNB-75	73.35 ± 1.98	80.34 ± 1.59
M14	71.40 ± 2.77	74.55 ± 2.24	SR	76.31 ± 1.80	81.74 ± 2.90
MALME-3M	72.71 ± 2.55	74.78 ± 2.16	SW-620	72.94 ± 2.38	74.82 ± 2.23
MCF7	74.95 ± 1.65	80.51 ± 2.53	T-47D	72.60 ± 2.45	75.55 ± 2.23
MDA-MB-231	70.91 ± 1.75	72.29 ± 3.15	TK-10	69.40 ± 2.87	67.42 ± 3.54
MDA-MB-435	71.85 ± 2.16	73.91 ± 2.07	U251	73.26 ± 1.53	75.83 ± 2.47
MDA-N	72.66 ± 2.39	74.13 ± 3.40	UACC-257	70.47 ± 1.91	71.86 ± 2.72
MOLT-4	76.21 ± 2.30	81.09 ± 1.45	UACC-62	73.98 ± 1.92	78.13 ± 2.50
NCI-ADR-RES	71.76 ± 2.73	73.21 ± 2.05	UO-31	72.16 ± 1.78	76.12 ± 2.18

Table 7
Precision and recall (and their standard deviations) obtained on the 60 NCI screens using k_{10}^{vt}

not allow paths to share edges once they have diverged. This degradation may be even stronger in the case of counts versus indicator variables. As we have seen, exhaustive search of all the paths up to depth d for reasonable values of d can easily be implemented given the low degree of molecular graphs. datasets used in the simulations reported here are 2.21 (Mutag), 2.03 (PTC), and 2.11 (NCI). Thus the typical degree for a compound in organic chemistry is below 2.3 giving a branching factor of less than 1.3, and a complexity factor of $O(n1.3^d)$ for searching all the paths up to length d.

Howeve, even when all paths of length up to d are considered, information is partially lost regarding their location (or "phase") within the molecular

Screen	Accuracy (%)	ROC (%)	Screen	Accuracy (%)	ROC (%)
786-0	72.62 ± 1.09	79.41 ± 0.89	NCI-H226	70.92 ± 1.15	77.53 ± 1.49
A498	71.81 ± 1.71	78.74 ± 1.93	NCI-H23	72.67 ± 1.24	79.27 ± 1.31
A549	72.38 ± 1.32	79.47 ± 1.46	NCI-H322M	70.60 ± 1.66	77.30 ± 1.32
ACHN	73.47 ± 1.78	80.47 ± 1.53	NCI-H460	72.45 ± 1.29	78.70 ± 1.08
BT-549	71.99 ± 1.62	79.39 ± 1.32	NCI-H522	73.89 ± 1.50	79.50 ± 1.71
CAKI-1	72.34 ± 0.88	79.30 ± 1.15	OVCAR-3	72.44 ± 1.29	79.19 ± 1.36
CCRF-CEM	72.36 ± 1.49	76.30 ± 1.72	OVCAR-4	72.64 ± 1.62	79.31 ± 1.39
COLO-205	72.67 ± 1.22	79.77 ± 1.40	OVCAR-5	70.44 ± 1.34	77.28 ± 1.63
DU-145	72.56 ± 1.32	80.18 ± 1.34	OVCAR-8	72.42 ± 1.44	78.82 ± 1.25
EKVX	71.41 ± 1.46	77.24 ± 1.63	PC-3	73.39 ± 1.69	80.66 ± 1.55
HCC-2998	70.62 ± 1.49	76.59 ± 1.32	RPMI-8226	73.02 ± 1.26	78.55 ± 1.57
HCT-116	74.01 ± 1.24	80.29 ± 1.18	RXF-393	71.48 ± 1.32	78.22 ± 1.15
HCT-15	71.87 ± 1.06	78.64 ± 1.24	SF-268	73.66 ± 1.32	79.29 ± 1.61
HL-60-TB	72.89 ± 1.45	76.48 ± 1.68	SF-295	71.62 ± 1.23	78.33 ± 1.13
HOP-62	72.06 ± 1.43	78.81 ± 1.58	SF-539	71.43 ± 1.48	77.29 ± 1.51
HOP-92	71.80 ± 1.41	77.50 ± 1.32	SK-MEL-28	70.89 ± 1.56	77.59 ± 1.47
HS-578T	72.27 ± 1.42	78.41 ± 1.72	SK-MEL-2	71.66 ± 1.47	78.88 ± 1.25
HT29	73.28 ± 1.70	79.50 ± 1.97	SK-MEL-5	74.05 ± 1.44	80.56 ± 1.26
IGROV1	71.03 ± 1.52	77.72 ± 1.51	SK-OV-3	71.83 ± 1.70	78.92 ± 1.58
K-562	71.51 ± 1.59	77.06 ± 1.80	SN12C	71.36 ± 1.27	78.31 ± 1.15
KM12	72.00 ± 1.54	78.84 ± 1.53	SNB-19	72.13 ± 1.07	79.40 ± 1.04
LOX-IMVI	73.59 ± 1.82	80.10 ± 1.64	SNB-75	70.32 ± 1.06	75.13 ± 1.39
M14	72.62 ± 2.08	79.73 ± 1.91	SR	73.32 ± 1.48	77.62 ± 1.46
MALME-3M	71.16 ± 1.41	78.11 ± 1.39	SW-620	72.76 ± 1.39	79.89 ± 1.37
MCF7	74.09 ± 1.74	80.60 ± 1.63	T-47D	73.08 ± 1.71	80.27 ± 1.78
MDA-MB-231	72.14 ± 1.34	79.07 ± 1.46	TK-10	71.17 ± 1.76	77.67 ± 1.70
MDA-MB-435	73.02 ± 1.98	80.07 ± 1.81	U251	73.60 ± 1.49	79.86 ± 1.66
MDA-N	72.81 ± 1.42	80.07 ± 1.35	UACC-257	70.91 ± 1.54	77.61 ± 1.25
MOLT-4	73.82 ± 1.62	78.52 ± 1.29	UACC-62	73.46 ± 1.83	79.70 ± 1.70
NCI-ADR-RES	72.02 ± 1.80	79.24 ± 1.25	UO-31	71.85 ± 0.95	78.06 ± 1.62

Table 8
Classification accuracy and ROC score (and their standard deviations) obtained on the 60 NCI screens using k_{10}^m

graphs or regarding their possible extensions to paths of length greater than d. Furthermore, none of the graph kernels considered so far contains information about the handedness of the molecules. Such information is not present in the datasets considered here and therefore could not have been used or evaluated. However it is clear that stereo chemical information is important– different isomers can have very different biological properties. Thus in time it will be useful to develop richer kernels, including kernels that are sensitive to sterochemical properties.

Several other molecular kernels can be considered using 2D graphs, but also several other molecular representations. In 2D, instead of extracting paths, one

Screen	Precision (%)	Recall (%)	Screen	Precision (%)	Recall (%)
786-0	73.19 ± 2.56	75.72 ± 1.44	NCI-H226	71.61 ± 1.86	72.86 ± 2.64
A498	72.15 ± 2.94	72.99 ± 2.29	NCI-H23	73.23 ± 2.05	76.56 ± 2.17
A549	71.42 ± 2.73	75.18 ± 2.01	NCI-H322M	69.25 ± 2.57	69.41 ± 3.14
ACHN	72.39 ± 2.05	76.37 ± 2.92	NCI-H460	74.48 ± 2.07	77.99 ± 1.66
BT-549	72.58 ± 2.11	73.20 ± 3.38	NCI-H522	76.77 ± 1.72	80.83 ± 1.73
CAKI-1	72.49 ± 1.43	75.33 ± 2.86	OVCAR-3	73.91 ± 1.70	76.31 ± 1.97
CCRF-CEM	76.28 ± 1.87	82.37 ± 1.66	OVCAR-4	73.04 ± 1.83	74.41 ± 2.36
COLO-205	73.72 ± 2.42	76.20 ± 2.47	OVCAR-5	66.95 ± 1.93	65.65 ± 3.25
DU-145	71.00 ± 2.45	72.64 ± 2.12	OVCAR-8	73.51 ± 2.56	76.01 ± 1.72
EKVX	72.21 ± 2.13	75.69 ± 2.93	PC-3	72.63 ± 1.95	77.05 ± 2.84
HCC-2998	72.19 ± 1.70	77.87 ± 2.39	RPMI-8226	76.01 ± 1.51	80.60 ± 1.76
HCT-116	75.06 ± 1.76	78.14 ± 3.26	RXF-393	72.96 ± 2.46	75.62 ± 1.95
HCT-15	72.67 ± 1.43	75.82 ± 2.47	SF-268	74.79 ± 1.87	78.24 ± 2.27
HL-60-TB	76.43 ± 1.89	83.48 ± 2.03	SF-295	73.50 ± 1.87	75.21 ± 2.68
HOP-62	72.42 ± 2.47	74.25 ± 2.44	SF-539	73.61 ± 2.32	78.05 ± 2.04
HOP-92	73.57 ± 1.71	77.80 ± 2.01	SK-MEL-28	69.11 ± 2.53	68.61 ± 2.57
HS-578T	72.98 ± 2.21	77.49 ± 1.68	SK-MEL-2	71.44 ± 2.55	70.57 ± 3.07
HT29	74.85 ± 2.44	76.61 ± 1.94	SK-MEL-5	75.15 ± 1.75	78.90 ± 2.32
IGROV1	71.16 ± 2.00	74.32 ± 2.40	SK-OV-3	71.13 ± 2.82	71.04 ± 3.15
K-562	74.22 ± 2.11	79.50 ± 2.58	SN12C	71.87 ± 2.08	73.64 ± 2.67
KM12	72.02 ± 2.43	76.57 ± 2.75	SNB-19	71.49 ± 1.70	73.15 ± 2.10
LOX-IMVI	75.34 ± 2.12	80.00 ± 2.24	SNB-75	73.36 ± 1.45	80.45 ± 2.22
M14	71.78 ± 2.65	75.31 ± 2.74	SR	76.87 ± 1.64	81.88 ± 1.83
MALME-3M	72.47 ± 2.26	75.39 ± 2.30	SW-620	73.48 ± 2.33	74.69 ± 2.36
MCF7	75.15 ± 1.79	81.38 ± 2.12	T-47D	73.00 ± 2.42	77.99 ± 2.40
MDA-MB-231	71.34 ± 1.93	74.43 ± 3.28	TK-10	69.46 ± 2.26	69.48 ± 3.29
MDA-MB-435	72.45 ± 2.56	75.87 ± 2.58	U251	74.59 ± 1.58	77.87 ± 2.85
MDA-N	73.12 ± 2.61	74.24 ± 3.25	UACC-257	71.07 ± 1.56	73.48 ± 3.13
MOLT-4	76.72 ± 2.42	82.22 ± 1.29	UACC-62	74.65 ± 2.17	79.07 ± 2.01
NCI-ADR-RES	71.97 ± 2.89	74.15 ± 2.28	UO-31	73.14 ± 1.84	76.93 ± 2.49

Table 9
Precision and recall (and their standard deviations) obtained on the 60 NCI screens using k_{10}^m

can focus on extracting breadth-first search and depth-first search spanning trees, whose computation is essentially the same as the one involved in the extraction of paths. Furthermore, as showed by Vishwanathan and Smola [2003], there exist very efficient ways to compare trees. In 1D, molecules can be represented as SMILES strings [James et al., 2004], to which the entire machinery of string kernels can be applied. In 3D, it is possible to construct kernels from the coordinates of all the atoms contained in a given molecule. While only a few hundred thousand molecules are in the Cambridge Structural Database (`http://www.ccdc.cam.ac.uk/products/csd`) of experimentally-determined structures, there exist programs, such as CORINA ([Sadowski et al., 1994,

Gasteiger et al., 1996]), that can derive full 3D coordinates for small organic molecules starting from their 1D or 2D representations. And in applications where molecular *surfaces* are the most important, "2.5D" kernels may be developed to characterize molecular surfaces. Recently, we have developed 1D kernels using SMILES strings, and 3D kernels using histograms of pairwise Euclidean distances between atom classes, and have compared them with the 2D kernels described here on the same datasets [Ralaivola et al., 2005]. In the long run, we can expect that different kernels may be designed for different tasks in computational chemistry and chemoinformatics.

Many other applications of both unsupervised and supervised machine learning methods to chemical data are possible. For instance, kernel methods could be applied to molecular clustering and regression problems, such as predicting the boiling point of alkanes or the QSAR (Quantitative Structure-Activity Relationship) of benzodiazepines [Cherqaoui and Villemin, 1994, Hadjipavlou-Litina and Hansch, 1994, Micheli et al., 2003].

More broadly, the penetration and adoption of artificial intelligence, statistics, and for that matter informatics methods in chemistry has been slower than in biology or physics, due to many factors including the single-investigator-nature of chemical research and the related absence of large-scale collaborative efforts in chemistry, in contrast to biology or physics, where genome sequencing or high-energy physics projects require coordination of hundreds of scientists. The lack of large, public, free-for-academic datasets of chemical information has also greatly hampered the development of robust informatics tools in chemistry. Similarity search tools such as BLAST have revolutionized biology. Similarity search tools, with kernels as one of their derivatives, are likelyd to play a similar role in chemistry as data on millions of compounds are beginning to become freely available [Jonsdottir et al., 2005, Baldi et al., 2005].

Large training datasets could be derived over time for a variety of problems in chemistry. However, critical improvements are required at the level of these datasets and their dissemination and annotation. Current annotated datasets, including those used in this work, are often small or biased and redundant. Coordinated, large-scale, annotation efforts, such as those commonly used in the biological sciences, would go a long way toward producing larger and better quality datasets for machine learning and other statistical approaches in chemistry. We hope that developing efficient kernels and other machine learning methods for molecular structures will foster over time greater synergies between bio- and chemical informatics [Dobson, 2004] and help us better understand chemical space.

Acknowledgment

Work supported by a Laurel Wilkening Faculty Innovation award, an NIH Biomedical Informatics Training grant (LM-07443-01), an NSF MRI grant (EIA-0321390), a Sun Microsystems award, a grant from the University of California Systemwide Biotechnology Research and Education Program to PB and an MD/PhD Harvey Fellowship to S. J. S. We would like also to acknowledge OpenEye Scientific Software for their free academic software license.

References

M. Aizerman, E. Braverman, and L. Rozonoér. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

F. R. Bach and M. I. Jordan. Kernel Independent Component Analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

P. Baldi, J. Bruand, J. Chen, Y. Dou, and S. J. Swamidass. ChemDB: a public, open source, database of small molecules. 2005. Submitted.

P. Baldi and S. Brunak. *Bioinformatics: the machine learning approach*. MIT Press, Cambridge, MA, 2001. Second edition.

P. Baldi and Y. Chauvin. Hybrid modeling, HMM/NN architectures, and protein applications. *Neural Computation*, 8(7):1541–1565, 1996.

P. Baldi and G. Pollastri. The principled design of large-scale recursive neural network architectures–DAG-RNNs and the protein structure prediction problem. *Journal of Machine Learning Research*, 4:575–602, 2003.

P. Baldi and M. Rosen-Zvi. On the relationship between deterministic and probabilistic directed graphical models: from Bayesian networks to recursive neural networks and back. *Neural Networks*, 2005. Special issue on Neural Networks and Kernel Methods for Structured Domains. In press.

B. Boser, I. Guyon, and V. Vapnik. A Training Algorithm for Optimal Margin Classifiers. In *Proc. of the 5th Workshop on Comp. Learning Theory*, volume 5, 1992.

D. Cherqaoui and D. Villemin. Use of neural network to determine the boiling point of alkanes. *J. Chem. Soc. Faraday Trans.*, 90:97–102, 1994.

P. Y. Chou and G. D. Fasman. Prediction of the secondary structure of proteins from their amino acid sequence. *Adv. Enzymol. Relat. Areas Mol. Biol.*, 47:45–148, 1978.

M. Collins and N. Duffy. Convolution Kernels for Natural Language. In *Adv. in Neural Information Processing Systems 14*, 2002. URL `citeseer.nj.nec.com/542061.html`.

C. Cortes and V. Vapnik. Support Vector Networks. *Machine Learning*, 20: 1–25, 1995.

N. Cristianini and J. Shawe-Taylor. Cambridge University Press, Cambridge, UK, 2000.

A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry*, 34:786–797, 1991.

C. M. Dobson. Chemical space and biology. *Nature*, 432:824–828, 2004.

S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proc. of the 7th Int. Conf. on Information and Knowledge Management*, 1998.

M. A. Fligner, J. S. Verducci, and P. E. Blower. A Modification of the Jaccard/Tanimoto Similarity Index for Diverse Selection of Chemical Compounds Using Binary Strings. *Technometrics*, pages 1–10, 2002.

D. R. Flower. On the properties of bit string-based measures of chemical similarity. *J. of Chemical Information and Computer Science*, 38:378–386, 1998.

P. Frasconi, M. Gori, and A. Sperduti. A general framework for adaptive processing of data structures. *IEEE Transactions on Neural Networks*, 9 (5):768–786, 1998.

Y. Freund and R. E. Schapire. Large Margin Classification Using the Perceptron Algorithm. *Machine Learning*, 37(3):277–296, 1999.

B. J. Frey. *Graphical Models for Machine Learning and Digital Communication*. MIT Press, 1998.

T. Friess, N. Cristianini, and N. Campbell. The Kernel-Adatron Algorithm: a Fast and Simple Learning Procedure for Support Vector Machines. In J. Shavlik, editor, *Proc. of the* 15^{th} *Int. Conf. on Machine Learning*. Morgan Kaufmann Publishers, 1998.

T. Gärtner. Exponential and Geomteric Kernels for Graphs. NIPS Workshop on Unreal Data: Principles of Modeling Nonvectorial Data, 2003.

T. Gärtner, P. A. Flach, and S. Wrobel. On Graph Kernels: Hardness Results and Efficient Alternatives. In *Proc. of the 16th Annual Conf. on Computational Learning Theory and 7th Kernel Workshop*, 2003.

J. Gasteiger, J. Sadowski, J. Schuur, P. Selzer, L. Steinhauer, and V. Steinhauer. Chemical information in 3D-space. *Journal of Chemical Information and Computer Sciences*, 36:1030–1037, 1996.

C. Goller and A. Kuchler. Learning task-dependent distributed structure-representations by backpropagation through structure. *IEEE International Conference on Neural Networks*, pages 347–352, 1996.

J. C. Gower. A general coefficient of similarity and some of its properties. *Biometrics*, 27:857–871, 1971.

J. C. Gower and P. Legendre. Metric and euclidean properties of dissimilarity coefficients. *Journal of Classification*, 3:5–48, 1986.

D. Hadjipavlou-Litina and C. Hansch. Quantitative structure-activity relationship of the benzodiazepines. A review and reevaluation. *Chemical Reviews*, 94:1483–1505, 1994.

J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

D. Haussler. Convolution Kernels on Discrete Structures. Technical Report UCS-CRL-99-10, UC Santa Cruz, 1999.

D. Heckerman. A tutorial on learning with Bayesian networks. In M.I. Jordan, editor, *Learning in Graphical Models*. Kluwer, Dordrecht, 1998.

C. Helma, R. D. King, S. Kramer, and A. Srinivasan. The predictive toxicology challenge 2000-2001. *Bioinformatics*, 17(1):107–108, 2001.

R. Herbrich. *Learning Kernel Classifiers, Theory and Algorithms*. MIT Press, 2002.

T. Horváth, T. Gärtner, and S. Wrobel. Cyclic pattern kernels for predictive graph mining. In *Proc. of the 2004 ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 158–167, 2004.

T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Adv. in Neural Information Processing Systems*, volume 11, 1998.

C. A. James, D. Weininger, and J. Delany. Daylight theory manual. 2004. Available at http://www.daylight.com/dayhtml/doc/theory/theory.toc.html.

S. O. Jonsdottir, F. S. Jorgensen, and S. Brunak. Prediction methods and databases within chemoinformatics: Emphasis on drugs and drug candidates. *Bioinformatics*, 2005.

H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized Kernels between Labeled Graphs. In *Proc. of the 20th International Conference on Machine Learning*, 2003.

G. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *J. Math. Anal. Applic.*, 33:82–95, 1971.

R. D. King, S. H. Muggleton, A. Srinivasan, and M. J. E. Sternberg. Structure-Activity Relationships Derived by Machine Learning: The Use of Atoms and their Bond Connectivities to Predict Mutagenicity by Inductive Logic Programming. *Proc. of the National Academy of Sciences*, 93(1):438–442, 1996.

R. D. King, A. Srinivasan, and M.J. E. Sternberg. Relating chemical activity to structure: an examination of ILP successes. *New Generation Computing*, 13:411–433, 1995.

R. I. Kondor and J. Lafferty. Diffusion Kernels on Graphs and other Discrete Input Spaces. In *Proc. of International Conference on Machine Learning*, 2002.

J.R. Koza. Evolution of a computer program for classifying protein segments as transmembrane domains using genetic programming. In R. Altman, D. Brutlag, P. Karp, R. Lathrop, and D. Searls, editors, *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, pages 244–252. AAAI Press, Menlo Park, CA, 1994.

S. Kramer and L. De Raedt. Feature construction with version spaces for biochemical application. In *Proc. of the 18th Int. Conf. on Machine Learning*,

pages 258–265, 2001.

S. L. Lauritzen. *Graphical Models*. Oxford University Press, Oxford, UK, 1996.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

C. Leslie, E. Eskin, A. Cohen, J. Weston, and W. S. Noble. Mismatch string kernels for SVM protein classification. In *Adv. in Neural Information Processing Systems*, volume 15, 2003.

C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for svm protein classification. In *Proc. of the Pacific Symposium on Biocomputing, 2002*, pages 564–575, 2002.

H. Lodhi, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text Classification using String Kernels. In *Adv. in Neural Information Processing Systems*, volume 15, 2000.

P. Mahé, N. Ueda, T. Akutsu, J.-L. Perret, and J.-P. Vert. Extension of Marginalized Graph Kernels. In *Proc. of the 21st Int. Conf. on Machine Learning*, New York, NY, USA, 2004.

A. Micheli, A. Sperduti, A. Starita, and A. M. Biancucci. A novel approach to QSPR/QSAR based on neural networks for structures. In H. Cartwright and L. M. Sztandera, editors, *Soft Computing Approaches in Chemistry*, pages 265–296. Springer Verlag, Heidelberg, Germany, 2003.

A. Micheli, A. Sperduti, A. Starita, and A. M. Bianucci. Analysis of the internal representations developed by neural networks for structures applied to quantitative structure-activity relationship studies of benzodiazepines. *J. Chem. Inf. Comput. Sci.*, 41:202–218, 2001.

S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. Kernel PCA and De-Noising in Feature Spaces. In *Adv. in Neural Information Processing Systems*, 1999. URL `citeseer.nj.nec.com/mika99kernel.html`.

S. Muggleton. volume 38 of APIC Series. Academic Press, London, 1992.

K.-R. Müller, G. Rätsch S. Mika, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Neural Networks*, 12(2):181–201, 2001.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA., 1988.

G. Pollastri, D. Przybylski, B. Rost, and P. Baldi. Improving the prediction of protein secondary strucure in three and eight classes using recurrent neural networks and profiles. *Proteins*, 47:228–235, 2001.

L. Ralaivola, J. Chen, J. Bruand, P. Phung, S. J. Swamidass, and P. Baldi. Kernels for small molecules and the prediction of mutagenicity, toxicity, and anti-cancer activity. *Bioinformatics*, Supplement 1, 2005. Proceedings of the 2005 ISMB Conference. In press.

J. W. Raymond and P. Willett. Effectiveness of graph-based and fingerprint-based similarity measures for virtual screening of 2D chemical structure

databases. *Journal of Computer-Aided Molecular Design*, 16:59–71, 2001.

D. Rouvray. *Journal of Chemical Information and Computer Sciences*, 32(6): 580–586, 1992.

J. Sadowski, J. Gasteiger, and G. Klebe. Comparison of automatic three-dimensional model builders using 639 X-ray structures. *Journal of Chemical Information and Computer Sciences*, 34:1000–1008, 1994.

Y. Sakakibara, M. Brown, R. Hughey, I. S. Mian, K. Sjölander, R. C. Underwood, and D. Haussler. Stochastic context-free grammars for tRNA modeling. *Nucl. Acids Res.*, 22:5112–5120, 1994.

G. Salton. Developments in automatic text retrieval. *Science*, 253:974–980, 1991.

B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. Technical Report NC-TR-00-081, NeuroCOLT, 2000.

B. Schölkopf and A. J. Smola. *Learning with Kernels, Support Vector Machines, Regularization, Optimization and Beyond*. MIT University Press, 2002. URL `http://www.learning-with-kernels.org`.

A. Sperduti and A. Starita. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–735, 1997.

A. Srinivasan, S. Muggleton, R. D. King, and M. Sternberg. Theories for mutagenicity: a study of first-order and feature based induction. *Artificial Intelligence*, 85:277–299, 1996.

K. Tsuda, T. Kin, and K. Asal. Marginalized kernels for biological sequences. *Bioinformatics*, 18, 2002.

A. Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, 1977.

E. Ukkonen. On–line construction of suffix trees. *Algorithmica*, 14:249–60, 1995.

V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, inc., 1998.

J.-P. Vert. A tree kernel to analyze phylogenetic profiles. *Bioinformatics*, 18, 2002.

J.-P. Vert and M. Kanehisa. Graph-driven features extraction from microarray data using diffusion kernel and kernel cca. In *Adv. in Neural Information Processing Systems*, volume 15, 2003.

S. V. N. Vishwanathan and A. J. Smola. Fast Kernels for Strings and Tree Matching. In *Adv. in Neural Information Processing Systems*, volume 15, 2003.

P. Weiner. Linear Pattern Matching Algorithms. In *Proc. of the 14th IEEE Ann. Symp. on Switching and Automata Theory*, pages 1–11, 1973.

Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proc. of the 14th Int. Conf. on Machine Learning*, pages 412–420, 1997. URL `citeseer.ist.psu.edu/yang97comparative.html`.

C. Yanover and Y. Weiss. Approximate inference and protein-folding. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 1457–1464. MIT Press, Cambridge, MA, 2003.